

Jena Research Papers in Business and Economics

A general solution framework for component commonality problems

Nils Boysen, Armin Scholl

12/2008

Jenaer Schriften zur Wirtschaftswissenschaft

Working and Discussion Paper Series
School of Economics and Business Administration
Friedrich-Schiller-University Jena

ISSN 1864-3108

Publisher:

Wirtschaftswissenschaftliche Fakultät
Friedrich-Schiller-Universität Jena
Carl-Zeiß-Str. 3, D-07743 Jena
www.jbe.uni-jena.de

Editor:

Prof. Dr. Hans-Walter Lorenz
h.w.lorenz@wiwi.uni-jena.de
Prof. Dr. Armin Scholl
armin.scholl@wiwi.uni-jena.de

www.jbe.uni-jena.de

A general solution framework for component commonality problems

Nils Boysen^{a*}, Armin Scholl^b

^a*Friedrich-Schiller-Universität Jena, Lehrstuhl für Allgemeine Betriebswirtschaftslehre/
Operations Management, Carl-Zeiß-Straße 3, D-07743 Jena, Germany,
nils.boysen@uni-jena.de*

^{*}*Corresponding author, phone +49 3641 943-100.*

^b*Friedrich-Schiller-Universität Jena, Lehrstuhl für Allgemeine Betriebswirtschaftslehre/
Betriebswirtschaftliche Entscheidungsanalyse, Carl-Zeiß-Straße 3, D-07743 Jena,
Germany, armin.scholl@wiwi.uni-jena.de*

Abstract

Component commonality, the use of the same version of a component across multiple products, is increasingly considered as a promising way to offer high external variety while retaining low internal variety in operations. However, increasing commonality has both positive and negative cost effects, so that optimization approaches are required to identify an optimal commonality level. As a more or less of components influences nearly every process step along the supply chain, it is not astounding that a multitude of diverging commonality problems is investigated in literature, each of which developing a specific algorithm designed for the respective commonality problem considered. The paper on hand aims at a general framework, flexible and efficient enough to be applied to a wide range of commonality problems. Such a procedure basing on a two-stage graph approach is presented and tested. Finally, flexibility of the procedure is shown by customizing the framework to account for different types of commonality problems.

Keywords: Product variety; Component commonality; Optimization; Graph approach

1 Introduction

In the recent years, firms more and more face the necessity of providing an enlarged product variety, which nowadays seems inevitable to successfully serve highly diversified customer demands. For instance, some car series especially from the luxury segment exceed billions of different car models (e.g., Boysen et al., 2008). In view of such an enormous variety, component commonality, the use of the same version of a component across multiple products (Fisher et al., 1999), is increasingly considered as a promising way to offer high external variety while retaining low internal variety in operations, and thus to lower cost (e.g., Swaminathan, 2001). In this context, a firm has to solve the basic decision problem of how many and what kinds of components to utilize. The degrees of freedom for such a component commonality problem range from providing a unique component for each single product up to a single component shared by all products (and any other solution in-between both extremes).

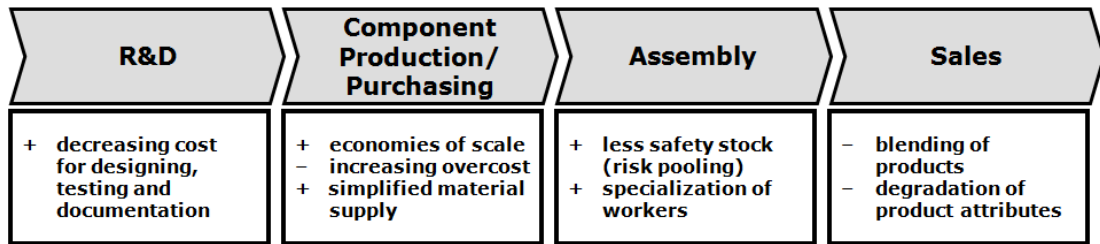


Figure 1: Impact of increasing component commonality along the supply chain

The extent of component commonality influences (nearly) any process step along the supply chain (see Figure 1). In R&D, any additional component needs to be designed, tested and documented, and thus increases cost (e.g., Fisher et al., 1999). Moreover, if commonality is increased and fewer components in a larger quantity are to be produced (or purchased), economies of scale can be realized (e.g., due to a reduced number of set-ups and orders (Tallon, 1989) as well as intensified learning (Thonemann and Brandeau, 2000)) and material supply to the final assembly is facilitated (Boysen et al., 2007). On the other hand, if multiple products share a common component, this component must meet specifications of the most demanding product, so that less discerning products receive a more valuable component than required (so called overcost, see Briant and Naddef, 2004). To decouple component production and final assembly, safety stocks need to be held, which can be reduced in size in case of increasing commonality due to risk pooling (e.g., Collier, 1982; Baker et al., 1986). During final assembly, less components reduce the variability of operations for the workforce (Perera et al., 1999). Finally in sales, commonality of visible components results in a blending of products, so that products become more indistinguishable from one another (e.g., Fisher et al., 1999). However, a negative impact also threatens from invisible components (e.g. a car battery) because indirectly product attributes might be degraded (e.g. increasing fuel consumption), see Ulrich (1995). Figure 1 summarizes the aforementioned effects of component commonality along the supply chain, where positive and negative consequences of an increasing

commonality are marked by “+” and “–”, respectively.

Figure 1 depicts just a brief excerpt of the cost effects of common parts discussed in literature. More exhaustive reviews are provided, e.g., by Ramdas (2003), Swaminathan and Lee (2004) as well as Labro (2004). With regard to the variety of different relationships between component commonality and supply chain operations it is not astounding that a massive body of literature has accumulated. Three major streams of research can be identified (see Labro, 2004): (i) inventory and operations related commonality research, (ii) R&D and engineering related commonality research and (iii) marketing related commonality research. Any of these streams covers a specific extract of the overall problem and any stream by itself contains a multitude of different research papers investigating specific component commonality problems. Consequently, plenty different solution approaches have been introduced, any of which being dedicated to the respective commonality problem treated. The paper on hand aims at a general framework for solving component commonality problems, which is both efficient and flexible enough to cover a multitude of different settings. For this purpose a two-stage graph approach is introduced, which can easily be customized for a specific commonality problem by simply changing the function to calculate arc weights in the graph.

The remainder of the paper is organized as follows. First, Section 2 provides a literature review on component commonality. Then, Section 3 identifies a general core problem of component commonality, which is formalized by a mathematical program. The solution framework is presented in Section 4 and initially described in solving the core problem of Section 3. Solution performance of this setting is tested in a comprehensive computational study in Section 5. Then, Section 6 shows how the solution framework can be adopted to cover extended versions of commonality problems taken from literature. Finally, Section 7 concludes the paper.

2 Literature review

As was already mentioned above, literature on component commonality can be separated into three streams of research (see Labro, 2004):

Inventory and operations related research: Dating back in the 1980s, component commonality was initially investigated with regard to its influence on inventories and operations. A multitude of different models, e.g., provided by Collier (1982), McClain et al. (1984), Baker (1985), Baker et al. (1986), Gerchak et al. (1988), Eynan and Rosenblatt (1996), Thonemann and Brandeau (2000), Hillier (2000, 2002) and Ma et al. (2002), consider the benefits of common parts, which are, for instance, a decrease in order/setup and inventory cost due to the risk pooling effect. On the one hand, setup cost are lowered by reducing the number of components as larger demands allow for larger lot sizes. On the other hand, having less components reduces the risk of forecast errors so that safety stock of such a multi-use component needs not be as large as the sum of safety stocks of the covered specialized components. This effect is referred to as risk pooling, because additional demand for one product and reduced demand for another one using the same component might compensate each other.

R&D and engineering related research: Later on, commonality research more and more shifted focus from inventory and operations aspects to R&D and engineering considerations. One argument might be that the majority of operation cost is already determined during the engineering phase (e.g., Swift et al., 2004) and another that commonality is especially employed in a make-to-order environment where inventory aspects are negligible (see Jans et al., 2008). Nevertheless, some models intermix engineering and inventory related aspects (e.g., Dogramaci, 1979; Thomas, 1991). R&D especially benefits from common parts by avoiding duplicate development cost (Fisher et al., 1999; Perera et al., 1999). Dogramaci (1979), Krishnan et al. (1999), Ramdas and Sawhney (2001) and Ramdas et al. (2003) provide models for commonality problems where fixed cost for component development are a major element of the total cost function. Engineering related commonality research typically restricts its models to the subset of components, which remain invisible to the customers (i.e., braking systems, Fisher et al., 1999, and Ramdas et al., 2003, or wiring harnesses, Thonemann and Brandeau, 2000).

Marketing related research: Finally, if components visible to the customer are standardized, commonality has also an influence on sales, i.e., customer preferences are met less precisely, which, in the last, decreases revenue. Research on this aspect of commonality stems, e.g., from Kim and Chhajed (2000), Desai et al. (2001) as well as Heese and Swaminathan (2006). In an industry case presented by Jans et al. (2008), prices are calculated on a cost-plus basis, so that cost consequences of component commonality indirectly influence revenues via the products' price elasticity.

The so-called assortment problem, which has a long lasting tradition of more than five decades (see Hanssman, 1957; Sadowski, 1959), can be seen as a forerunner of commonality research. For an extensive review on this problem see Pentico (2008). The assortment problem considers downward substitutability of products with just a single (significant) feature. As cost components, overcost and fixed cost for component development are to be minimized. Although the assortment problem was initially dedicated to stocking situations, it can be applied to a wide range of related situations, one of which is component commonality. However, merely simple cost structures and just a single feature are considered, which hinders a direct application of the assortment problem in real-world commonality problems. The relationship of both fields of research are discussed in detail by Pentico (2008).

From a methodological point of view, commonality research mainly utilizes analytical models (e.g., Collier, 1982; McClain et al., 1984; Baker et al., 1986; Gerchak et al., 1988; Desai et al., 2001; Hillier, 2000, 2002; Ma et al., 2002) to gain general insights, nevertheless, also a wide arsenal of algorithmic optimization approaches is applied in literature to act as decision support in determining an optimal level of commonality. Plenty exact procedures have been developed, i.e., mathematical programming (Briant and Naddef, 2004; Jans et al., 2008), dynamic programming (Sadowski, 1959; Rutenberg, 1971), branch&bound (Thonemann and Brandeau, 2000). Furthermore, a lot of heuristic approaches are introduced in literature, i.e., clustering methods (Dogramaci, 1979; Thomas, 1991), priority rules (Gupta and Krishnan, 1999), simulated annealing

(Thonemann and Brandeau, 2000), and decomposition approaches (Avella et al., 2005). Any of these procedures was designed to cover a specific component commonality problem, whereas our solution framework is flexible and efficient enough to be applied to a wide range of commonality problems. Moreover, our framework is able to act both as an exact and a heuristic solution procedure.

3 Description of a basic component commonality problem

In this section, a basic component commonality problem is developed, which exemplifies the elementary trade-off and exhibits all basic properties of more general component commonality problems. By means of this basic problem version, the general course of our solution framework is described and solution performance is tested in Sections 4 and 5, respectively.

A given set P of products with a given demand $d_p \forall p \in P$ is to be provided with components of a specific kind. Each product p has (minimum) requirements to be fulfilled by its designated component. These requirements refer to a set F of features owned by a component. Any feature $f \in F$ can receive different values $v \in V_f$, so that fixing a single value for each feature composes a complete specification of a component. Thus, a component commonality problem has to answer three interrelated questions: (i) How many components with (ii) what specification to select and (iii) which product to provide with which component? To clarify our nomenclature the following example of automobile industry is given: Different car models (products) are to be supplied with sunroofs (component). A major property of sunroofs is the drive (feature), which might be manually (value 1) or electrically (value 2) powered.

Furthermore, it is assumed that values $v \in V_f$ per feature f are sorted in increasing order according to their ability to meet products' requirements, so that a value v per feature f is able to also fulfill a requirement for another value v' of the same feature, if $v' < v$ holds, but not vice versa. Literature on commonality labels this property as downward compatibility or one-way substitutability. For our example, this would mean that any customer would except an electrical sunroof, if his/her minimum standard is a manual sunroof, but not the other way round. The minimum requirement value (some $v \in V_f$) of a product p with regard to feature f is denoted by the parameter r_{pf} .

In our basic commonality problem, we only consider two kinds of cost. On the one hand, fixed cost K occur whenever an additional component is introduced and, e.g., represents all cost for developing, testing and documenting a component. On the other hand, unit cost of the components are captured, which originate from the realized specification of the respective component and are calculated by cumulating cost k_{fv} of the actually realized value v over all features f .

Example: We consider $|P| = 5$ products with minimum requirements r_{pf} as given in Table 1. Each of the $|F| = 3$ features has just the single value 1, so that $V_f = \{1\} \forall f \in F$. So, in each case, the requirement is 0 (feature f is not needed by product p) or 1 (product p needs the unique value of feature f). Thus, product 1 needs none of the features, while

| p | r_{pf} | | | d_p |
|----------|----------|---------|---------|----------|
| | $f = 1$ | $f = 2$ | $f = 3$ | |
| 1 | 0 | 0 | 0 | 10 |
| 2 | 1 | 0 | 0 | 20 |
| 3 | 0 | 0 | 1 | 10 |
| 4 | 0 | 1 | 1 | 20 |
| 5 | 1 | 1 | 1 | 10 |
| k_{f1} | 1 | 1 | 1 | $K = 20$ |

Table 1: Example data

product 5 needs them all. Fixed cost K amount to 20 money units. Any other data of our component commonality problem is listed in Table 1, too. A possible solution for this example would be to introduce three components that serve products 1 and 2, 3 and 4 and product 5, respectively. The component for products 1 and 2 merely contains feature $f = 1$ and is to be produced $\sum_{p=1}^2 d_p = 30$ times, so that variable and fixed cost amount to $k_{11} \cdot \sum_{p=1}^2 d_p + K = 1 \cdot 30 + 20 = 50$. Since product 1 does not need the feature, overhead cost of $d_1 \cdot k_{11} = 10$ have to be paid for exceeding the requirement of product 1. This, however, spares paying the fixed component cost of $K = 20$. The overall cost D for the aforementioned solution result to $D = 180$, which is the optimal solution for this problem instance.

| | |
|-----------|---|
| P | set of products (index p) |
| C | set of components (index c) |
| F | set of features (index f) |
| V_f | set of values per feature f (index v) |
| k_{fv} | cost to realize value v of feature f |
| K | fixed cost per component |
| r_{pf} | minimum requirement (some value $v \in V_f$) of product p with regard to feature f |
| d_p | demand for product p |
| y_c | binary variables: 1, component c is introduced; 0, otherwise |
| x_{pc} | binary variables: 1, product p receives component c ; 0, otherwise |
| z_{cfv} | binary variables: 1, component c realizes value v of feature f ; 0, otherwise |

Table 2: Notation

In the decision model the specification of a component c is denoted by binary variables z_{cfv} , which receive value 1, whenever value v of feature f is realized in c (0, otherwise). The assignment of products $p \in P$ to components $c \in C$ is covered by binary variables x_{pc} , which are assigned value 1, if product p receives component c (0, otherwise). The binary variables y_c indicate whether a component c is actually chosen for production (1) or not (0). As the components are constructed within the model via the variables z_{cfv} , it is not necessary to enumerate the set of possible components (which contains

$\prod_{f \in F} (|V_f| + 1)$ elements). However, in order to restrict the number of variables to be defined in the model prior to computation, we use the simple insight that at most $|P|$ components are required in a solution. This maximal number would be obtained in the extreme case of doing without any component commonality.

With the help of the notation summarized in Table 2 the basic core component commonality problem (CCCP) consists of objective function (1) and constraints (2) to (6):

$$(\text{CCCP}) \text{ Minimize } D(X, Y, Z) = \sum_{c \in C} \sum_{f \in F} \sum_{v \in V_f} z_{cfv} \cdot k_{fv} \cdot \sum_{p \in P} x_{pc} \cdot d_p + \sum_{c \in C} y_c \cdot K \quad (1)$$

subject to

$$\sum_{c \in C} x_{pc} = 1 \quad \forall p \in P \quad (2)$$

$$x_{pc} \leq y_c \quad \forall p \in P; c \in C \quad (3)$$

$$\sum_{v \in V_f} z_{cfv} \leq 1 \quad \forall c \in C; f \in F \quad (4)$$

$$x_{pc} \cdot r_{pf} \leq \sum_{v \in V_f} z_{cfv} \cdot v \quad \forall p \in P; c \in C; f \in F \quad (5)$$

$$y_c, x_{pc}, z_{cfv} \in \{0, 1\} \quad \forall p \in P; c \in C; f \in F; v \in V_f \quad (6)$$

In objective function (1) total cost are to be minimized, which consist of variable cost (first term) and fixed cost (second term). Variable cost are calculated by multiplying unit cost per component, which is cumulated over the contained feature values, by the demand of those products that receive the respective component. Equations (2) ensure that each product receives exactly one component, whereas constraints (3) enforce that, if a component is assigned to a product ($x_{pc} = 1$) the component is to be introduced ($y_c = 1$), so that respective fixed cost accrue in the objective function. Furthermore, it is to be ensured by inequalities (4) that each component can realize at most one value per feature. Finally, constraints (5) enforce that minimum requirements of products are met. Whenever a component c is assigned to a product p ($x_{pc} = 1$), then, the requirement r_{pf} of the product is to be satisfied by at least the same realized value or an even better one.

The CCCP is NP-hard (see Briant, 2000). This property becomes obvious, if the downward compatibilities between products and their feature requirements r_{pf} are transferred to a directed graph (see Briant and Naddef, 2004). Each product receives a node in the graph (plus additional “virtual” products representing all additional non existent combinations of feature values). If a component designed to meet minimum requirements of a product i can also be integrated in a less demanding (real) product j , an arc (j, i) with arc weight $w_{ji} = \sum_{f \in F} k_{f, r_{if}} \cdot d_j$ is inserted. Any other arc required to build a complete graph is assigned an arc weight with a prohibitive large value. With this graph on hand CCCP becomes equivalent to an uncapacitated facility location problem (UFLP; e.g., Klose and

Drex1, 2005), where opening a facility represents introducing a component (connected with fixed cost K) and delivery cost are equivalent to total variable cost of component production. The UFLP is well known to be NP-hard (see Karup and Pruzan, 1983). Note that a special case of CCCP with just a single feature can be solved in polynomial time, since the problem becomes an assortment problem with one-way substitutability (see Rutenberg, 1971), which can be solved, e.g., with the famous Wagner-Whitin algorithm (Wagner and Whitin, 1958) for dynamic single item lotsizing (see Sadowski, 1959).

4 Solution framework

4.1 General procedure

The solution framework bases on a decomposition of the overall problem into two stages and resembles the solution procedure of Boysen and Flidner (2008) for assembly line balancing:

- First stage, one or more different orders of products are determined and stored in a set of sequence vectors.
- These product sequences are passed over to the second stage, where given orders of products are translated to a directed graph, which is applied to determine groups of products jointly served by a component and is, thus, labeled grouping graph. Once a grouping graph is constructed, solving the component commonality problem (for the set of given product sequences) reduces to finding the shortest path in the grouping graph.

The general idea of this solution framework bases on the following consideration. If products are ordered and stored in a sequence vector π , any possible grouping of products can be evaluated by a simple shortest-path-approach, provided that the following grouping policy is obeyed: Only products, which are adjacent to each other in the product sequence π and, thus, form a subsequence of π may be unified to a product group. To allow for an intuitive understanding of this policy before the graph approach is formally described, Figure 2 displays a grouping graph for the given product order of $\pi = \langle 2, 3, 1 \rangle$.

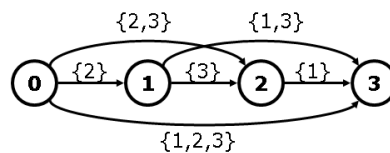


Figure 2: Example grouping graph for a given product order of $\pi = \langle 2, 3, 1 \rangle$

As depicted in Figure 2 any possible grouping of products (represented by arcs and the sets of products stored with each arc) is contained in the graph, except for the product group $\{1, 2\}$, which would violate our grouping policy (products 1 and 2 are separated by product 3 within sequence π). If, furthermore, arc weights can be determined, which

represent the cost associated with a component designed for the product set represented by the arc, then, solving the CCCP reduces to finding the shortest-path from source node 0 to the respective sink node. The length of the shortest-path equals the optimal total cost $D^*(\pi)$ for a given order π .

A grouping graph can also be constructed for multiple product sequences, so that, at the first stage, one or more promising orders of products need to be determined. A detailed and formal description of this drafted general idea is provided in the following subsections, where both stages are described in reverted order as this facilitates comprehension.

4.2 Stage 2: Grouping graph

Input of Stage 2 is a set Π of sequences $\pi_i \in \Pi$ with $i = 1, \dots, |\Pi|$, each of which representing an order of products, so that products are stored at sequence positions $\pi_i(s)$, with $s = 1, \dots, |P|$. This input is applied to construct the grouping graph, which is defined as digraph $G = (V, E, c)$ composed of node set V , arc set E and an arc weighting function $c : E \rightarrow \mathbb{R}$, respectively.

The overall node set V is subdivided into $s = 1, \dots, |P|$ stages plus an additional start node 0. Each stage s represents a sequence position and is assigned a subset $V_s \subseteq V$ of nodes. A node i of stage s denoted by $i(s)$ represents an occurrence of products in a sequence π_i up to position s . The respective product set P_{is} of sequence i is defined as follows: $P_{is} = \{\pi_i(s') \mid s' = 1, \dots, s\}$. Even different sequences might lead to identical subsets of products considered up to position s . To avoid additional computational effort for a duplicate inspection of identical nodes and associated product sets, only unique nodes $i(s)$ with regard to their product set P_{is} are generated:

$$V_s = \{i(s) \mid i = 1, \dots, |\Pi| : P_{is} \neq P_{js} \quad \forall j = 1, \dots, i-1\} \quad \forall s = 1, \dots, |P| \quad (7)$$

Note that avoiding duplicate product sets leads to a single node 1($|P|$) in final stage $s = |P|$, because any (feasible) order of products contains all products up to the final stage, so that: $P_{i,|P|} = P \forall i = 1, \dots, |\Pi|$. The stage dependent node sets V_s (plus initial start node 0) are unified to the overall node set V :

$$V = \bigcup_{s=1}^{|P|} V_s \cup \{0\} \quad (8)$$

Two nodes $i(s)$ and $j(s')$ are connected by an arc, if the following conditions hold: (i) node $j(s')$ belongs to a later stage than node $i(s)$, so that $s < s'$ holds and (ii) product set P_{is} of node $i(s)$ is a subset of product set $P_{js'}$ belonging to node $j(s')$:

$$E = \{(i(s), j(s')) \mid s = 1, \dots, |P| - 1, s' = s + 1, \dots, |P|, i \in V_s, j \in V_{s'} : P_{is} \subseteq P_{js'}\} \quad (9)$$

Each arc represents a single component, which is dedicated to a special subset of products. This subset PS_{ij} of products assigned to an arc $(i(s), j(s'))$ is equal to the difference set, i.e., $PS_{ij} = P_{js'} \setminus P_{is}$. Set PS_{ij} is stored with each arc and contains all products jointly served by the same component. This graph structure is a general element of the solution framework and remains unaltered irrespective of the specific component commonality problem actually investigated.

An arc weight represents the total cost of introducing the represented component. Consequently, its calculation depends on the specific cost structures of the respective commonality problem and is, thus, the basic element to customize our solution framework for a specific problem. In case of our basic commonality CCCP, an arc weight c_{ij} of an arc connecting nodes $i(s)$ and $j(s')$ receives variable cost VC_{ij} and fixed cost K : $c_{ij} = VC_{ij} + K \forall (i, j) \in E$, where variable cost VC_{ij} are calculated as follows:

$$VC_{ij} = \left(\sum_{f \in F} k_{fv_f^*} \right) \cdot \left(\sum_{p \in PS_{ij}} d_p \right) \quad \forall (i, j) \in E \quad (10)$$

The index $v_f^* = \max\{r_{pf} | p \in PS_{ij}\}$ denotes the value of the highest requirement per feature $f \in F$ of all assigned products from set PS_{ij} . With the help of index v_f^* the respective cost $k_{fv_f^*}$ per feature f can be identified, cumulated over all features and weighted with the overall demand of assigned products. How to adopt the calculation of arc weights to solve variational component commonality problems is discussed in Section 6.

With such a grouping graph on hand, a component commonality problem reduces to finding the shortest path from the unique source node 0 with product set $P_0 = \emptyset$ to the unique sink node $1(|P|)$ with an assigned product set of $P_{1|P|} = P$. The length of the shortest path equals the minimum total cost $D^*(\Pi)$ for the given set of product sequences Π .

Note that the graph approach can also be applied if only a single product order ($|\Pi| = 1$) is determined at the first stage. However, as arcs allow for a cross over between different product sequences, the solution value D^* obtained by a unified grouping graph for a given set of product orders with $|\Pi| > 1$ is always better or equal to a successive examination of isolated sequences $\pi_i \in \Pi$: $D^*(\Pi) \geq \min \{D^*(\pi_i) | i = 1, \dots, |\Pi|\}$. This property is demonstrated by the following example.

Example: Given the problem instance of Table 1 and two product sequences $\pi_1 = \{1, 2, 3, 5, 4\}$ and $\pi_2 = \{1, 3, 2, 4, 5\}$. If a separate grouping graph is constructed for both sequences both solutions amount to an overall cost $D^*(\pi_1) = D^*(\pi_2) = 190$. The resulting two separated grouping graphs along with their bold faced shortest paths are depicted in Figure 3. However, if a unique grouping graph is built for both sequences the overall optimal solution for the CCCP-instance with total cost of $D^*(\Pi) = 180$ is identified (see Figure 4).

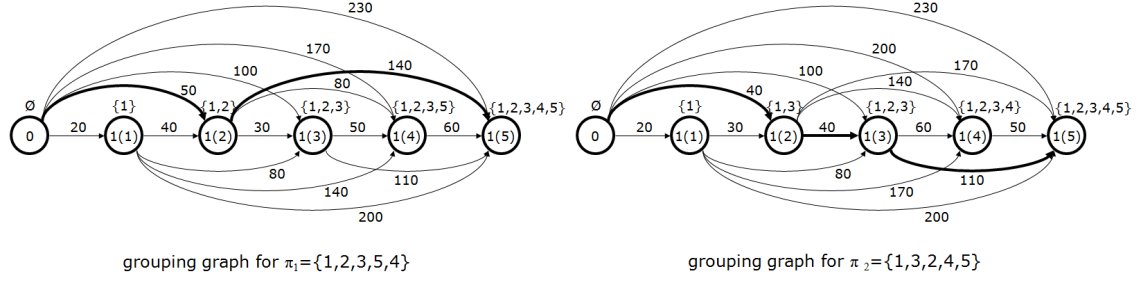


Figure 3: Separate grouping graphs for given product sequences

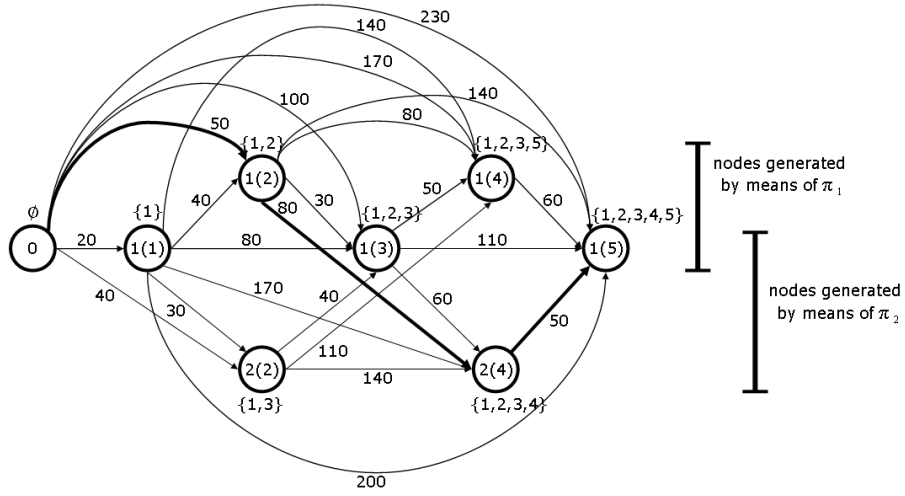


Figure 4: Unique grouping graph for the example

4.3 Stage 1: Sequencing of products

There exist numerous alternatives of how to determine adequate product sequences. These alternatives can, i.e., be classified by the number of product sequences generated:

- If *all* possible successions of products are generated and passed over to Stage 2, obviously the overall optimal solution D^* is determined and our approach acts as an exact solution procedure. However, such a complete enumeration suffers from the extraordinary number of possible sequences, which is $\frac{|P|!}{2}$. Thus, the ability of our solution framework to act as an exact solution procedure is more a theoretical one, especially if problem instances of real-world size are to be solved. Anyhow, this property is useful, if heuristic settings of our framework are to be evaluated according to their solution quality.
- On the other hand, only a *single* sequence can be produced. In this case, computational effort is reduced for the price of solution quality. One possibility would be to adopt a binary sorting procedure, which is, e.g., often applied to the so called cell-formation-problem in Group Technology (see King and Nakornchai, 1982; Burbidge, 1991). This problem deals with forming groups of products, which are jointly produced in a separate shop and require similar resources to reduce investment cost. To adopt binary sorting, all features are to be resorted in ascending order according to the following priority value w_f , where v^* is the maximum number of different values per feature (including absence of the feature, i.e., value 0) over all products: $v^* = \max\{r_{pf} + 1 | p \in P; f \in F\}$:

$$w_f = \sum_{p \in P} r_{pf} \cdot (v^*)^{(|P|-p)} \quad \forall f \in F \quad (11)$$

Finally, the resulting reordered requirements matrix \mathbf{r} is applied to determine an initial product sequence π according to the following priority value u_p in descending order:

$$u_p = \sum_{f \in F} r_{pf} \cdot (v^*)^{(|F|-f)} \quad \forall p \in P \quad (12)$$

This procedure resorts the requirements matrix \mathbf{r} , so that “blocks” of similar requirements can be identified (exemplified by Figure 5). According to this block structure Equations (12) assign each product p a priority value u_p .

Example (cont.): The resulting product sequence is $\pi = \{5, 2, 4, 3, 1\}$. The optimal grouping for this sequence, $\{5\}, \{2\}, \{4\}, \{1, 3\}$, obtained by the grouping graph results in total cost of $D^*(\pi) = 190$.

- A compromise between both extremes would be to produce *some* solutions. A very simple advancement would be to approach a random sampling and to determine a number x of randomly drawn sequences. A more sophisticated approach to identify

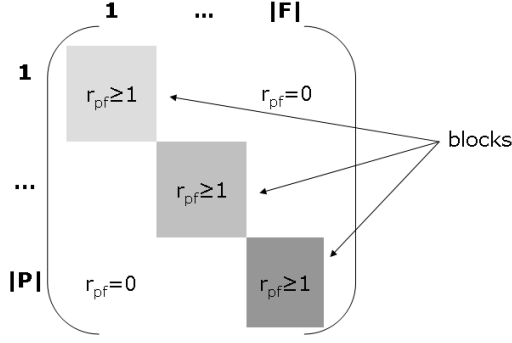


Figure 5: Intended block structure after binary sorting of requirements matrix \mathbf{r}

a promising subset of product sequences would be to apply a meta heuristic. In the following, an Ant Colony approach (see Dorigo et al., 1999) is developed.

In an Ant Colony approach, solutions are constructed repetitively by software agents (artificial ants), which typically base their decisions on some local heuristic measure and the collected experiences of all former ants, aggregated in a so called pheromone matrix. The search process of an individual ant resembles a simple priority rule based heuristic, such that at each sequence position s a single product is chosen out of the set POS_s of possible alternatives (products not yet scheduled). An ant's sequence π_i is hence filled from left to right. However, the choices of an ant are not deterministic, but stochastic according to a weighted probability scheme which is repetitively calculated at each decision point (sequencing position). The probability $Prob(p, s)$ that product p is assigned to position s is then determined on the basis of its priority value $w(p, p^{s-1})$ and the intensity of the pheromone $\tau_{pp^{s-1}}$ with respect to its alternatives, where p^{s-1} is the previously scheduled product in the sequence $p^{s-1} = \pi_i(s-1)$:

$$Prob(p, s) = \frac{\tau_{pp^{s-1}}^\alpha \cdot \left(\frac{1}{w(p, p^{s-1})} \right)^\beta}{\sum_{p' \in POS_s} \tau_{p'p^{s-1}}^\alpha \cdot \left(\frac{1}{w(p', p^{s-1})} \right)^\beta} \quad \forall s = 2, \dots, |P|; p \in POS_s \quad (13)$$

As priority value $w(p, p^{s-1})$ we simply measure the similarity between the previously scheduled product p^{s-1} and candidate product p according to priority value w_p of equation (12): $w(p, p^{s-1}) = |w_{p^{s-1}} - w_p| \quad \forall p \in POS_s$. Analogously, pheromone value $\tau_{pp^{s-1}}$ is determined between predecessor p^{s-1} and actual product p , so that pheromone is stored in a $|P| \times |P|$ -matrix. The initial product of each ant's sequence is randomly drawn. Parameters α and β control the relative importance of the pheromone versus the priority values. Because of experiences with other sequencing problems reported in the literature, these parameters are set to $\alpha = 1$ and $\beta = 2$ (see Stützle and Dorigo, 1999).

In this way, all ants belonging to the actual iteration k construct their respective sequence. Once all $|\Pi|$ sequences are generated, this set of sequences is passed over to Stage 2, where the grouping graph is constructed and the best product grouping for the iteration is determined. Note that each stage's grouping graph is discarded after having determined the respective solution. This way computational effort for constructing additional arcs is restricted for the price of losing information about promising groupings. The optimal solution of iteration k can be retranslated into an optimal sequence $\pi(k)$, which along with the corresponding solution value $D^*(\pi(k))$ is applied to update the pheromone trail. Thus, pheromone value $\tau_{pp'}(k)$ in iteration k is calculated as follows:

$$\tau_{pp'}(k) = \tau_{pp'}(k-1) \cdot (1-\rho) + \rho \cdot \begin{cases} \frac{1}{D^*(\pi(k))}, & \text{if } \pi(k, s) = p \wedge \pi(k, s-1) = p' \\ 0, & \text{otherwise} \end{cases} \quad \forall p, p' \in P \quad (14)$$

The formula incorporates two mechanisms for guiding the search. Older pheromone is constantly reduced (evaporation) which strengthens the influence of more recent solutions and new pheromone is assigned to all product successions, which are part of the solution, in proportion to the respective objective value. The parameter ρ , which is set to 0.5, controls the relative importance of these two components. Note that the pheromone matrix has to be initialized with starting values $\tau_{pp'}(0) = \frac{1}{D^*(\pi_{start})} \forall p, p' \in P$, where π_{start} represents a first, randomly drawn product sequence. In the current implementation 20 ants are employed to construct solutions in any iteration. After 500 iterations the algorithm terminates and the best solution found is returned.

Which alternative of sequence generation is an appropriate choice mainly depends on the computational effort a planner is willing to spend. A more detailed answer can be stated with the help of the computational study in the following section.

5 Computational study

Up to now, commonality research exclusively investigates different special problem settings mostly inspired from real-world cases. Consequently, no established test bed for our basic commonality problem CCCP is available. Therefore, we first elaborate on the instances that are used in our computational study. Then, experimental results on the performance of algorithms are presented.

5.1 Instance generation

In our computational study, we distinguish between two classes of test instances: small and large instances. The small instances are designed such that our solution framework can still solve all test instances to optimality (in acceptable time). Large instances shall represent problem instances of a size relevant in real world settings, where only heuristic

| symbol | description | values | |
|-------------|---|-------------------|-----------------------|
| | | small | large |
| $ P $ | number of products | $5, 6, \dots, 10$ | $75, 100, \dots, 200$ |
| $ F $ | number of features | $3, 4, \dots, 7$ | |
| V_f^{max} | maximum number of (non-zero) values per feature | 4 | |
| K | fixed cost for component development | 5000 | |

Table 3: Parameters for instance generation

solutions are obtainable. To derive these instance classes the input parameters listed in Table 3 are used to produce the requirements of products r_{pf} , product demands d_p and variable cost k_{fv} per feature and value defining a CCCP-instance.

Within each test case, these parameters are combined in a full-factorial design and instance generation per parameter constellation is repeated 10 times, so that $2 \cdot 6 \cdot 5 \cdot 10 = 600$ different CCCP-instances are obtained. On the basis of a given set of parameters each single instance is generated as follows:

- *Product requirements:* First, the number of values V_f per feature f are randomly determined by drawing an uniformly distributed integer out of the interval $[1, V_f^{max}]$. Then, the products' requirements r_{pf} are fixed by randomly drawing an uniformly distributed integer out of the interval $[0, V_f]$ in each case.
- *Product demands:* The demands d_p of products p are randomly drawn with uniform distribution out of the interval $[1, 1000]$.
- *Variable cost:* Finally, a feature specific real value ϱ^f , which is the basic cost factor per feature f , is randomly drawn (with uniform distribution) out of interval $[0.5, 1.5]$. This factor is applied to determine variable cost k_{vf} per value v of feature f : $k_{fv} = \varrho^f \cdot v \ \forall f \in F; v = 1, \dots, V_f$.

All generated instances can be downloaded from the internet at the research homepage for assembly system optimization (<http://www.assembly-line-balancing.de>).

5.2 Performance of algorithms

All methods have been implemented in C# (Visual Studio 2003) and run on a Pentium IV, 1800 MHz PC, with 512 MB of memory. First, the performance of the procedures with regard to the small instances is evaluated (see Table 4). These instances can be solved to optimality by a complete enumeration (labeled "ALL") of all possible product sequences (only reverted sequences of already generated once can be left out, see Section 4.3). For this exact procedure, we report the average solution time, measured in CPU-seconds and abbreviated by "avg cpu". Compared to optimal objective values, solution performance for our solution framework is reported if the priority rule approach (PRIO),

a random sampling (RAND) of 20 sequences and our Ant Colony approach (ANTS) is applied in the first stage, respectively. To capture solution performance, Table 4 lists the average (maximum) relative deviation from the optimum (labeled “avg gap” (“max gap”)) in percent for any parameter constellation, where deviations per instance are measured by: $\frac{D(x)-D(ALL)}{D(ALL)} \cdot 100\% \forall x \in \{PRIO, RAND, ANTS\}$.

Table 4 reveals an exponential increase of solution time required to determine an optimal solution with increasing number $|P|$ of products. This result is not astounding because the number of sequences to be evaluated increases exponentially in $|P|$, as well. On the other hand, solution times of PRIO and RAND are negligible as within neither instance more than 0.1 CPU-seconds are required. Both heuristic approaches, PRIO and RAND, show very promising results as the average gap over all instances amounts to merely 1.2% and 0.7%, respectively. According to the trade-off between solution time and quality, ANTS ranges in between. It solves a remarkable number of 273 instances (93%) to optimality with an average gap of merely 0.1% at an average computational time of 1.8 CPU-seconds.

Table 5 lists the results for the large test instances. Here, optimal solutions remain unknown, so that the quality measures, “avg gap” and “max gap”, are calculated in relation to the best objective value obtained per instance by one of the three procedures, PRIO, RAND and ANTS. To reasonably restrict computational time, the number of iterations of ANTS is bound to 20 with 5 ants generating sequences per iteration, whereas RAND is executed in an unchanged setting generating 20 random product sequences.

With regard to solution quality, ANTS shows superior. It contributes the best objective value to any instance except for three (where RAND finds the best solution). However, ANTS requires a considerable amount of computational time with an average of 115 CPU-seconds. With $|P| = 200$ products the maximum computational time is 363 CPU-seconds. Thus, instances with $|P| > 200$ can only be solved by ANTS if even more computational time is accepted or the number of iterations and ants is further reduced. Again, PRIO requires least computational time with an average of only 0.4 CPU-seconds. What is even more, the average gap amounts to merely 1.6%. Finally, RAND shows not competitive, as it is inferior with regard to both time and quality compared to PRIO. In instances of real-world size the solution space seems far to large, so that in contrast to the small instances a random sampling is not able to cover a sufficient proportion of all possible product sequences. Consequently, our priority rule based approach (PRIO) seems best suited for generating near optimal solutions, whenever instances of real-world size are to be solved in a very short time frame.

6 Customizing the solution framework

In this section, different extensions of our basic commonality problem CCCP are investigated of how to customize our solution framework. These extensions are subdivided according to the aforementioned classification of Labro (2004) into inventory, engineering and marketing related issues of component commonality.

| $ P $ | $ F $ | ALL | PRIO | | RAND | | ANTS | | |
|-------|-------|---------|---------|---------|---------|---------|---------|---------|---------|
| | | avg cpu | avg gap | max gap | avg gap | max gap | avg gap | max gap | avg cpu |
| 5 | 3 | 0.02 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.4 |
| | 4 | 0.02 | 0.8 | 5.7 | 0.0 | 0.1 | 0.0 | 0.0 | 0.4 |
| | 5 | 0.02 | 1.5 | 11.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.4 |
| | 6 | 0.02 | 1.2 | 5.9 | 0.0 | 0.0 | 0.0 | 0.0 | 0.5 |
| | 7 | 0.02 | 1.5 | 12.9 | 0.0 | 0.0 | 0.1 | 0.8 | 0.5 |
| 6 | 3 | 0.10 | 0.2 | 0.9 | 0.0 | 0.0 | 0.1 | 0.8 | 0.7 |
| | 4 | 0.09 | 0.9 | 4.7 | 0.0 | 0.0 | 0.0 | 0.0 | 0.7 |
| | 5 | 0.10 | 1.1 | 3.7 | 0.3 | 2.6 | 0.0 | 0.0 | 0.7 |
| | 6 | 0.10 | 0.7 | 5.0 | 0.2 | 1.7 | 0.0 | 0.0 | 0.8 |
| | 7 | 0.10 | 2.2 | 5.7 | 0.8 | 4.3 | 0.0 | 0.0 | 0.8 |
| 7 | 3 | 0.67 | 0.9 | 5.5 | 0.6 | 3.9 | 0.0 | 0.0 | 1.0 |
| | 4 | 0.68 | 0.1 | 0.8 | 0.3 | 2.6 | 0.0 | 0.0 | 1.1 |
| | 5 | 0.68 | 1.0 | 3.4 | 0.3 | 2.7 | 0.0 | 0.0 | 1.2 |
| | 6 | 0.67 | 0.6 | 1.9 | 0.9 | 4.5 | 0.0 | 0.0 | 1.3 |
| | 7 | 0.68 | 1.6 | 7.1 | 0.5 | 3.9 | 0.0 | 0.0 | 1.3 |
| 8 | 3 | 5.52 | 0.2 | 1.5 | 0.3 | 2.8 | 0.0 | 0.0 | 1.5 |
| | 4 | 5.46 | 1.2 | 4.8 | 0.7 | 3.8 | 0.0 | 0.0 | 1.7 |
| | 5 | 5.44 | 0.7 | 2.8 | 1.2 | 7.5 | 0.1 | 0.5 | 1.8 |
| | 6 | 5.45 | 1.4 | 5.0 | 1.1 | 4.4 | 0.2 | 1.5 | 1.9 |
| | 7 | 5.45 | 2.3 | 6.1 | 1.0 | 3.2 | 0.1 | 0.7 | 2.1 |
| 9 | 3 | 50.8 | 0.9 | 8.5 | 0.2 | 2.5 | 0.0 | 0.0 | 2.2 |
| | 4 | 51.0 | 0.4 | 2.0 | 0.5 | 2.7 | 0.0 | 0.3 | 2.4 |
| | 5 | 50.8 | 1.8 | 8.2 | 0.6 | 2.5 | 0.0 | 0.0 | 2.8 |
| | 6 | 50.7 | 1.5 | 4.8 | 0.7 | 3.4 | 0.2 | 1.8 | 3.0 |
| | 7 | 50.7 | 1.9 | 5.9 | 1.3 | 2.4 | 0.1 | 0.7 | 3.2 |
| 10 | 3 | 558.3 | 1.3 | 6.2 | 1.9 | 7.1 | 0.0 | 0.0 | 3.3 |
| | 4 | 540.6 | 0.9 | 3.1 | 2.0 | 4.8 | 0.1 | 1.4 | 3.6 |
| | 5 | 536.8 | 2.7 | 6.5 | 0.9 | 2.4 | 0.3 | 2.8 | 4.0 |
| | 6 | 542.5 | 2.7 | 6.7 | 2.4 | 4.2 | 0.2 | 1.2 | 4.1 |
| | 7 | 534.6 | 2.3 | 5.9 | 1.4 | 2.7 | 0.2 | 1.9 | 4.7 |
| total | | 99.9 | 1.2 | 12.9 | 0.7 | 7.5 | 0.1 | 2.8 | 1.8 |

Table 4: Performance of procedures for small instances

| $ P $ | $ F $ | PRIO | | | RAND | | | ANTS | | |
|-------|-------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| | | avg gap | max gap | avg cpu | avg gap | max gap | avg cpu | avg gap | max gap | avg cpu |
| 75 | 3 | 2.3 | 4.0 | <0.1 | 10.9 | 18.0 | 4 | 0.0 | 0.0 | 11 |
| | 4 | 2.1 | 3.9 | 0.1 | 5.5 | 10.2 | 4 | 0.0 | 0.0 | 13 |
| | 5 | 2.4 | 4.0 | 0.1 | 6.2 | 10.2 | 5 | 0.0 | 0.0 | 15 |
| | 6 | 1.9 | 3.2 | 0.1 | 4.5 | 6.2 | 5 | 0.0 | 0.0 | 17 |
| | 7 | 1.9 | 2.6 | 0.1 | 3.6 | 4.6 | 6 | 0.0 | 0.0 | 19 |
| 100 | 3 | 1.5 | 3.2 | 0.1 | 8.5 | 15.4 | 7 | 0.0 | 0.0 | 25 |
| | 4 | 1.7 | 2.8 | 0.1 | 8.3 | 15.4 | 8 | 0.0 | 0.0 | 31 |
| | 5 | 1.6 | 3.4 | 0.1 | 5.5 | 9.6 | 9 | 0.0 | 0.0 | 35 |
| | 6 | 1.6 | 2.1 | 0.1 | 4.6 | 8.3 | 11 | 0.0 | 0.0 | 40 |
| | 7 | 2.2 | 3.7 | 0.2 | 5.0 | 9.4 | 12 | 0.0 | 0.0 | 45 |
| 125 | 3 | 2.0 | 4.9 | 0.2 | 12.6 | 22.7 | 12 | 0.0 | 0.0 | 52 |
| | 4 | 2.0 | 3.4 | 0.2 | 8.7 | 15.9 | 14 | 0.0 | 0.0 | 60 |
| | 5 | 1.7 | 3.9 | 0.2 | 7.3 | 11.6 | 16 | 0.0 | 0.0 | 68 |
| | 6 | 1.9 | 2.8 | 0.3 | 5.2 | 8.2 | 18 | 0.0 | 0.0 | 77 |
| | 7 | 1.4 | 2.2 | 0.3 | 4.8 | 9.4 | 20 | 0.0 | 0.0 | 87 |
| 150 | 3 | 0.5 | 2.3 | 0.5 | 15.8 | 30.5 | 20 | 0.0 | 0.0 | 87 |
| | 4 | 1.2 | 3.0 | 0.5 | 8.9 | 16.1 | 23 | 0.0 | 0.0 | 103 |
| | 5 | 1.3 | 2.8 | 0.6 | 6.2 | 9.6 | 26 | 0.0 | 0.0 | 119 |
| | 6 | 1.3 | 1.9 | 0.7 | 6.0 | 9.8 | 29 | 0.0 | 0.0 | 133 |
| | 7 | 1.5 | 2.2 | 0.8 | 4.7 | 7.7 | 32 | 0.0 | 0.0 | 148 |
| 175 | 3 | 1.0 | 3.3 | 0.5 | 12.1 | 22.9 | 30 | 0.0 | 0.0 | 139 |
| | 4 | 1.7 | 3.0 | 0.5 | 8.7 | 11.3 | 33 | 0.0 | 0.0 | 161 |
| | 5 | 1.6 | 3.6 | 0.6 | 6.6 | 9.4 | 29 | 0.0 | 0.0 | 186 |
| | 6 | 1.6 | 2.8 | 0.7 | 4.7 | 6.7 | 41 | 0.0 | 0.0 | 206 |
| | 7 | 1.4 | 3.1 | 0.8 | 4.2 | 7.1 | 44 | <0.1 | 0.2 | 228 |
| 200 | 3 | 0.6 | 2.6 | 0.7 | 14.1 | 19.4 | 40 | 0.0 | 0.0 | 201 |
| | 4 | 1.6 | 2.7 | 0.8 | 7.5 | 10.2 | 45 | <0.1 | 0.3 | 233 |
| | 5 | 1.4 | 2.8 | 0.9 | 6.2 | 8.9 | 51 | 0.0 | 0.0 | 257 |
| | 6 | 1.3 | 2.1 | 1.0 | 5.6 | 8.8 | 57 | 0.0 | 0.0 | 302 |
| | 7 | 1.4 | 2.1 | 1.2 | 4.6 | 6.0 | 64 | 0.0 | 0.0 | 339 |
| total | | 1.6 | 4.9 | 0.4 | 7.2 | 30.5 | 24 | <0.1 | 0.3 | 115 |

Table 5: Performance of procedures for large instances

6.1 Inventory and operations related extensions

Inventory and setup cost: Whenever components are produced to stock, inventory cost accrue and a reorder policy needs to be applied. Inspired by a real-world commonality problem of wiring harnesses, Thonemann and Brandeau (2000) model a continuous review (R_c, Q_c) policy. That is, whenever the stored quantity of a component $c \in C$ reaches the reorder point R_c , a new order of quantity Q_c is placed. Delivery requires a constant lead time τ . Additionally, it is assumed that for each component a fill rate of β should be guaranteed (β -service level), i.e., $\beta \cdot 100\%$ of all orders have to be fulfilled directly from stock for all components. The demands of products $p \in P$ are assumed to be independent random variables with expected demand rates d_p (average demand per period) and standard deviations σ_p for the cumulated demand during the replenishment lead time τ . These parameters are used to define expected demand rates μ_{ij} and standard deviations σ_{ij} of lead time demands for all components, which are represented by arcs $(i, j) \in E$ within our solution framework (see Section 4):

$$\mu_{ij} = \sum_{p \in PS_{ij}} d_p \quad \forall (i, j) \in E \quad (\text{expected demand of components}) \quad (15)$$

$$\sigma_{ij} = \sqrt{\sum_{p \in PS_{ij}} (\sigma_p)^2} \quad \forall (i, j) \in E \quad (\text{std. dev. of demand in lead time}) \quad (16)$$

Unit inventory holding cost rates h_{ij} per time unit for any component are computed by multiplying expected variable cost (measuring the economic value added, i.e., the capital locked) with a constant interest rate \bar{h} (see Equations (10)):

$$h_{ij} = \bar{h} \cdot VC_{ij} \quad \forall (i, j) \in E \quad (17)$$

The order quantities Q_{ij} and reorder points R_{ij} are approximated as follows with S denoting the fixed setup cost incurring each time an order is placed for any component, $\Psi(z) = \int_{t=z}^{\infty} (t-z)d\Phi(t)$ denoting the standard loss function and $\Phi(\cdot)$ denoting the cumulated distribution function of a standard normal variate:

$$Q_{ij} = \sqrt{\frac{2 \cdot \mu_{ij} \cdot S}{h_{ij}}} \quad \forall (i, j) \in E \quad (18)$$

$$R_{ij} = \tau \cdot \mu_{ij} + \sigma_{ij} \cdot \Psi^{-1} \left(\frac{(1-\beta) \cdot Q_{ij}}{\sigma_{ij}} \right) \quad \forall (i, j) \in E \quad (19)$$

The expected inventory holding cost per time unit HC_{ij} are approximated as sum of two cost components, the inventory holding cost for all safety stocks, HC_{ij}^1 , and the inventory holding cost for regular stock, HC_{ij}^2 , as follows (for details see Thonemann und Brandeau, 2000):

$$HC_{ij}^1 = h_{ij} \cdot \sigma_{ij} \cdot \Psi^{-1} \left(\frac{(1-\beta) \cdot Q_{ij}}{\sigma_{ij}} \right) \quad \forall (i, j) \in E \quad (20)$$

$$HC_{ij}^2 = h_{ij} \cdot \frac{Q_{ij}}{2} \quad \forall (i, j) \in E \quad (21)$$

$$HC_{ij} = HC_{ij}^1 + HC_{ij}^2 \quad \forall (i, j) \in E \quad (22)$$

The expected setup or order cost per time unit is computed by summing up the expected order cost of all components which are derived from dividing the setup cost factor S by the time between orders (fraction of order quantity and expected demand rate):

$$SC_{ij} = \frac{S \cdot \mu_{ij}}{Q_{ij}} \quad \forall (i, j) \in E \quad (23)$$

Depending on their relevance in a component commonality setting, inventory cost HC_{ij} and setup cost SC_{ij} can be added to other cost components like our fixed and variable cost of CCCP altogether building the cost per component and, thus, arc weights c_{ij} within our solution framework. The additional cost components considered by Thonemann and Brandeau (2000) can also be covered by our solution framework. Their production cost equal the variable cost of the CCCP model and so-called “complexity cost” can be considered with an extension presented in Section 6.2 for the case of nonlinear increasing fixed cost. Consequently, our solution framework can be applied for the complete commonality problem defined by Thonemann and Brandeau (2000), which is the most general one in existing literature, without difficulty.

Decreasing variable cost due to learning: An increase of component commonality entails that remaining components are produced in larger quantity (at least under the premise that commonality does not affect product sales) and, thus, economies of scale can be realized. Although Thonemann and Brandeau (2000) as well as Jans et al. (2008) state that learning is an important influencing factor in many real-world commonality problems, it has not been covered by commonality research, thus far. However, learning can be easily incorporated in our solution framework.

For instance, the elementary power model proposed by Wright (1936) assumes the following learning curve:

$$k_n = k_1 \cdot n^{-b} \quad (24)$$

where k_x , n and b denote production cost in the x^{th} cycle, number of cycles and learning constant, respectively. By building the integral of (24) and rearranging the term total production cost K_n over all n cycles amount to (see Dar-El, 2000, Sec. 3.1.1):

$$K_n = \left(k_1 \cdot n^{(1-b)} \right) \cdot \frac{1}{1-b} \quad (25)$$

With this formula on hand, the total production cost depending on the degree of component commonality can be calculated and assigned to an arc (i, j) as its arc weight c_{ij} , where v_f^* denotes the maximum value of feature f of all assigned products, see Equations (10):

$$c_{ij} = \left(\left(\sum_{f \in F} k_{fv_f^*} \right) \cdot \left(\sum_{p \in PS_{ij}} d_p \right)^{(1-b)} \right) \cdot \frac{1}{1-b} + K \quad \forall (i, j) \in E \quad (26)$$

Analogously, other learning models (e.g., see Yelle, 1979; Dar-El, 2000) can be integrated, if they base on (i) initial production cost and (ii) total volume of production, because this information can be readily determined with the help of the data stored with each arc in our solution framework.

6.2 R&D and engineering related extensions

Incompatibilities: An important issue during R&D are incompatibilities between certain values of different features. For instance, a subassembly to realize value v of feature f might obstruct the installation slot of another value v' of another feature f' . If these incompatibilities are not considered during the engineering phase, infeasible component specifications might result. Thus, if existent, incompatibilities need to be considered in component commonality problems. A simple advancement would be to exclude all arcs from the graph, whose assigned product sets lead to component specifications requiring incompatible feature values. However, this is only a heuristic because an upgrade of a subset of values to fix incompatibilities might be less costly than excluding the respective grouping of products. Among all feasible upgrades of values the least costly is to be identified to maintain the property of our solution framework of being able to be applied as an exact approach. If the solution framework is applied with only a subset of product sequences (which is the usual choice for commonality problems of real-world size) and serves as a heuristic, excluding the respective arcs keeps the solution framework simple.

Nonlinear increase of fixed cost: In CCCP fixed cost for component development increase linear in the number of components. This assumption is often not fulfilled in real-world commonality problems (see Thonemann and Brandeau, 2000). For instance, some empirical studies reveal an inverted learning curve with an increasing number of components (e.g., see Wildemann, 1994, p. 367). To account for arbitrary functions f of fixed cost K depending on the number of components $|C|$: $K = f(|C|)$, a special shortest path procedure needs to be applied. This procedure is an adoption of the approach of Saigal (1968), which determines the shortest path with a given number of k arcs applied. In our modified approach, first, all shortest paths with k arcs for any possible arc number $k = 1, \dots, |P|$ are determined, where only variable cost are considered as arc weights. Then, fixed cost $f(k)$ for k components are added to any of $|P|$ shortest paths determined and the minimum over these solutions is the overall optimal solution for a given set of sequences. The additional notation required is summarized in Table 6. A formal description is as follows:

- (1) Determine the structure of the grouping graph, where fixed cost are excluded from calculating arc weights. If nonlinear fixed cost are the only alteration compared to base

| | |
|----------|--|
| k | number of arcs applied in a path between two nodes |
| V^+ | Set of nodes without start node 0: $V^+ = V \setminus \{0\}$ |
| $R_j(k)$ | length of the shortest path to node j among all paths with k arcs |
| $P_j(k)$ | ordered sequence of nodes on the shortest path to node j among all paths with k arcs |

Table 6: Additional notation for nonlinear fixed cost

model CCCP arc weights equal variable cost VC_{ij} (see 10): $c_{ij} = VC_{ij} \forall (i, j) \in E$.

- (2) Initialize the following data: $R_j(1) = c_{0j} \forall j \in V^+$; $P_j(1) = \langle 0, j \rangle \forall j \in V^+$; $k:=1$.
- (3) Recursively determine length and nodes on the shortest path to any node $j \in V^+$ where the number of arcs is restricted to exactly k : $R_j(k+1) = \min \{R_i(k) + c_{ij} \mid (i, j) \in E\} \forall j \in V^+$ and $P_j(k+1) = \langle P_{i^*}(k), j \rangle \forall j \in V^+$, where i^* is the respective predecessor node on the shortest path and $\langle P, j \rangle$ denotes that element j is appended to list P .
- (4) Set $k := k + 1$ and goto Step 3, unless $k > |P|$.
- (5) Add nonlinear fixed cost (represented by function $f(k)$) to any shortest path. The minimum cost over all k solutions is the minimum overall solution for the given set of sequences: $D^* = \min \{R_{|V^+|}(k) + f(k) \mid k = 1, \dots, |P|\}$.

6.3 Marketing related extensions

Finally, component commonality has also an important influence on the market side (see Sections 1 and 2). Although, some analytical papers on the relationship between component commonality and sales have been published in the recent years, the paper of Jans et al. (2008) was the first to integrate sales aspects in an optimization model on commonality. In their industrial case study, a given set of products (power tools) needs to be partitioned in families. Within each family all products receive a common stage and frame, where the one product per family having the largest engine determines the requirements for both components (downward substitutability). As hitherto, in such a setting component commonality influences the variable cost of the common components (stage and frame) for each family, whereas additional groups entail fixed cost for component development. Additionally, prices for the power tools are calculated on a cost-plus basis, so that unit cost are simply increased by a given percentage mark-up. Then, sales are anticipated with the help of a given price elasticity when compared to the old selling price, so that the net present value of resulting returns can be maximized. As percentage mark-up, price elasticity and old selling price per product are all given parameters the solution of the Jans et al. problem only depends on the grouping of products and can, thus, be easily solved with our solution framework.

7 Conclusion

The paper on hand introduces a two stage graph approach, which is flexible enough to be applied to a wide range of component commonality problems. The solution performance of the procedure was shown to be very promising if applied to a basic component commonality problem. If similar results can be obtained for a broader class of component commonality problems cannot be answered at present due to the apparent lack of benchmark problems and comparable procedures. As the shortest path problems in the second stage are always solved to optimality (except for incompatibilities, see Section 6.2), irrespective of the considered extensions, the ability of identifying promising product sequences will most likely have the strongest impact on the solution quality. Our computational study revealed that our solution framework shows robust solution quality irrespective of the first stage procedure, so that it can be expected that this will hold for the vast majority of presented extensions alike. However it remains up to future research to further support this conjecture.

Another promising field for future commonality research would be to consider the interrelationship between different components. On the one hand, a technical interrelationship might be relevant, whenever, for instance, commonality leads to some heavier multi-purpose components, which altogether would exceed a given maximum weight allowed for the final product or increase its energy demand. On the other hand, component commonality leads to a blending of products in the customer's perception, which, however, can be compensated with other exceptional properties (components) of the respective product. Thus, all types of components and decisions on their levels of commonality are interrelated with regard to the customer's utility valuation of the products. To explicitly cover this effect, the advancement of relating component commonality via a cost-plus price setting and a price elasticity (see Jans et al., 2008, and Section 6.3) is not sufficient. Consequently, joint optimization models of product line selection (e.g. see Green and Krieger, 1985; Nair et al., 1995) and component commonality are required to capture the overall decision problem in a more detailed fashion.

References

- [1] Avella, P., Boccia, M., Di Martino, C., Oliviero, G., Sforza, A., Vasil'ev, I., 2005. A decomposition approach for a very large scale optimal diversity management problem, *4OR* 3, 23–37.
- [2] Baker, K.R., 1985. Safety stocks and component commonality, *Journal of Operations Management* 6, 13–22.
- [3] Baker, K.R., Magazine, M.J., Nuttle, H.L.W., 1986. The effect of commonality on safety stock in a simple inventory model, *Management Science* 32, 982–988.
- [4] Boysen, N., Fliedner, M., 2008. A versatile algorithm for assembly line balancing, *European Journal of Operational Research* 184, 39–56.

- [5] Boysen, N., Fliedner, M., Scholl, A., 2007. Level scheduling of mixed-model assembly lines under storage constraints, *International Journal of Production Research* (to appear).
- [6] Boysen, N., Fliedner, M., Scholl, A., 2008. Assembly line balancing: Joint precedence graphs under high product variety, *IIE Transactions* (to appear).
- [7] Briant, O., 2000. Etude théorique et numérique du problème de la gestion de la diversité. Ph.D. thesis, INP Grenoble, France. <ftp://ftp.imag.fr/pub/bibliotheque/theses/2000/Briant.Olivier/>.
- [8] Briant, O., Naddef, D., 2004. The optimal diversity management problem, *Operations Research* 52, 515–526.
- [9] Burbidge, J.L., 1991. Production flow analysis for planning group technology, *Journal of Operations Management* 10, 5–27.
- [10] Collier, D.A., 1982. Aggregate safety stock levels and component part commonality, *Management Science* 28, 1296–1303.
- [11] Dar-El, E.M., 2000. Human learning: From learning curves to learning organizations, Springer, Berlin.
- [12] Desai, P., Kekre, S., Radhakrishnan, S., Srinivasan, K., 2001. Product differentiation and commonality in design: Balancing revenue and cost drivers, *Management Science* 47, 37–51.
- [13] Dogramaci, A., 1979. Design of common components considering implications of inventory costs and forecasting, *AIIE Transactions* 11, 129–135.
- [14] Dorigo, M., Di Caro, G., Gambardella, L.M., 1999. Ant algorithms for discrete optimization, *Artificial Life* 5, 137–172.
- [15] Eynan, A., Rosenblatt, M.J., 1996. Component-commonality effects on inventory costs, *IIE Transactions* 28, 93–104.
- [16] Fisher, M., Ramdas, K., Ulrich, K., 1999. Component sharing in the management of product variety: A study of automotive braking systems, *Management Science* 45, 297–315.
- [17] Gerchak, Y., Magazine, M.J., Gamble, A.B., 1988. Component commonality with service level requirements, *Management Science* 34, 753–760.
- [18] Green, P.E., Krieger, A.M., 1985. Models and heuristics for product line selection, *Marketing Science* 4, 1–19.
- [19] Gupta, S., Krishnan, V., 1999. Integrated component and supplier selection for a product family, *Production and Operations Management* 8, 163–182.

- [20] Hanssmann, F., 1957. Determination of optimal capacities of service for facilities with a linear measure of inefficiency, *Operations Research* 5, 713–717.
- [21] Heese, H.S., Swaminathan, J.M., 2006. Product line design with component commonality and cost-reduction effort, *Manufacturing & Service Operations Management* 8, 206–219.
- [22] Hillier, M.S., 2000. Component-commonality in multiple-period, assemble-to-order systems, *IIE Transactions* 32, 755–766.
- [23] Hillier, M.S., 2002. Using commonality as backup safety stock, *European Journal of Operational Research* 136, 353–365.
- [24] Jans, R., Degreave, Z., Schepens, L., 2008. Analysis of an industrial component commonality problem, *European Journal of Operational Research* 186, 801–811.
- [25] Karup, J., Pruzan, P.M., 1983. The simple plant location problem: Survey and synthesis, *European Journal of Operational Research* 12, 36–81.
- [26] Kim, K., Chhajed, D., 2000. Commonality in product design: Cost savings, valuation change and cannibalization, *European Journal of Operational Research* 125, 602–621.
- [27] King, J.R., Nakornchai, V., 1982. Machine-component group formation in group technology: Review and extension, *International Journal of Production Research* 20, 117–133.
- [28] Klose, A., Drexl, A., 2005. Facility location models for distribution system design, *European Journal of Operational Research* 162, 4–29.
- [29] Krishnan, V., Singh, R., Tirupati, D., 1999. A model-based approach for planning and developing a family of technology based products, *Manufacturing & Service Operations Management* 1, 132–156.
- [30] Labro, E., 2004. The cost effects of component commonality: a literature review through a management-accounting lens, *Manufacturing & Service Operations Management* 6, 358–367.
- [31] Ma, S., Wang, W., Liu, L., 2002. Commonality and postponement in multistage assembly systems, *European Journal of Operational Research* 142, 523–538.
- [32] McClain, J.O., Maxwell, J.A., Muckstadt, L.J., Weiss, T.E.N., 1984. Comment on “Aggregate safety stock levels and component part commonality”, *Management Science* 30, 772–773.
- [33] Nair, S.K., Thakur, L.S., Wen, K.-W., 1995. Near optimal solutions for product line design and selection: Beam search heuristics, *Management Science* 41, 767–785.

- [34] Pentico, D.W., 2008. The assortment problem: A survey, *European Journal of Operational Research* 190, 95–309.
- [35] Perera, H.S.C., Nagarur, N., Tabucanon, M.T., 1999. Component part standardization: A way to reduce the life cycle costs of products, *International Journal of Production Economics* 60-61, 109–116.
- [36] Ramdas, K., 2003. Managing product variety: An integrative review and research directions, *Production and Operations Management* 12, 79–101.
- [37] Ramdas, K., Sawhney, M.S., 2001. A Cross-functional approach to evaluating multiple line extensions for assembled products, *Management Science* 47, 22–36.
- [38] Ramdas, K., Fisher, M., Ulrich, K., 2003. Managing variety for assembled products: Modeling component systems sharing, *Manufacturing & Service Operations Management* 5, 142–156.
- [39] Rutenberg, D.P., 1971. Design commonality to reduce multi-item inventory: Optimal depth of a product line, *Operations Research* 19, 491–509.
- [40] Sadowski, W., 1959. A few remarks on the assortment problem, *Management Science* 6, 13–24.
- [41] Saigal, R., 1968. A constrained shortest route problem, *Operations Research* 16, 205–209.
- [42] Stützle, T., Dorigo, M., 1999. ACO algorithms for the quadratic assignment problem. In: Corne, D., Dorigo, M., Glover, F. (Eds.), *New Ideas in Optimization*. McGraw-Hill, pp. 33–50.
- [43] Swaminathan, J.M., 2001. Enabling customization using standardized operations, *California Management Review* 43(3), 125–135.
- [44] Swaminathan, J.M., Lee, H.L., 2004. Design for postponement. In: Graves S.C., De Kok, A.G. (Eds.) *Handbooks in Operations Research and Management Science*, Vol. 11., *Supply Chain Management: Design, Coordination, and Operation*. Elsevier, pp. 199–226.
- [45] Swift, K.G., Booker, J.D., Edmondson, N.F., 2004. Strategies and case studies in assembly system selection, *Proceedings of the I MECH E Part B Journal of Engineering Manufacture* 218, 675–88.
- [46] Tallon, W.J., 1989. A comparative analysis of master production scheduling techniques for assembly-to-order products, *Decision Science* 20, 492–506.
- [47] Thomas, L.D., 1991. Commonality analysis using clustering methods, *Operations Research* 39, 677–680.

- [48] Thonemann, U.W., Brandeau, M.L., 2000. Optimal commonality in component design, *Operations Research* 48, 1–19.
- [49] Ulrich, K., 1995. The role of product architecture in the manufacturing firm, *Research Policy* 24, 419–440.
- [50] Wagner, H.M., Whitin, T.M., 1958. Dynamic version of the economic lot size model, *Management Science* 5, 89–96.
- [51] Wildemann, H., 1994. *Fertigungsstrategien*, 2.nd ed. München (in German).
- [52] Wright, T., 1936. Factors affecting the cost of airplanes, *Journal of Aeronautical Sciences* 3, 122–128.
- [53] Yelle, L.E., 1979. The learning curve: Historical review and comprehensive survey, *Decision Science* 10, 302–328.